KING SAUD UNIVERSITY
COLLEGE OF COMPUTER & INFORMATION SCIENCES
DEPT OF COMPUTER SCIENCE

CSC311 Computer Algorithms
Second Semester 1444 AH

Tutorial #4

By. 3meer

Problem 1

(a) Describe an in-place algorithm that takes as input an array of n integers and rearranges it so that all even integers come first and then all odd integers come next.
You are allowed to use ONLY a constant amount of extra storage.

```
procedure sortArray(array arr)
   left = 0                                          → 1
   right = length(arr) - 1                           → 1

   while (left < right)                              → n/2
      if (arr[left] is odd)                          → n/2
         swap(arr[left], arr[right])                 → n/2
      else if (arr[left] is even)                    → n/2
         left = left + 1                             → n/2

      if (arr[right] is even)                        → n/2
         right = right - 1                           → n/2
      end if
   end while

   return arr                                        → 1
end procedure

function swap(int a, int b)
   temp = a
   a = b
   b = temp
end function
```

(b) What is the time complexity of your algorithm? You need to show the step count and a Θ estimate.

Big-O

$\frac{7n}{2} + 3 \leq c\,g(n)$

$\leq 4n + 3n$

$\leq 7n$

$c_1 = 7 \quad n_0 = 1 \quad \therefore O(n)$

Big-Ω

$\frac{7n}{2} + 3 \geq c\,g(n)$

$\geq c \cdot n$

$n_0 = \frac{\sum_{i=0}^{k-1} |a_i|}{a_k - c} = \frac{3}{7-6} = 3$

$n_0 = 3 \quad c_2 = 6 \quad \therefore \Omega(n)$

Theta Θ

$c_1 = 7 \quad c_2 = 6 \quad n_0 = 3$

$\therefore \Theta(n)$

Problem 2

Solve the following instance of the Knapsack Problem using Dynamic programming paradigm. The maximum allowed weight is **W**max = **10**.

i	**1**	**2**	**3**	**4**
Vi	20	10	15	31
Wi	6	1	2	5

a- Give the recursive equation you used to define the data structure needed for your dynamic programming solution. Then, fill the proposed data structure.

Recursive equation :

$c[i, w] = \max(c[i-1, w], c[i-1, w - wi] + vi)$

i \ w	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	20	20	20	20	20
2	0	10	10	10	10	10	20	30	30	30	30
3	0	10	15	25	25	25	25	30	35	45	45
4	0	10	15	25	25	31	41	46	56	56	56

b- What is your solution to this instance of the Knapsack problem?

$s = \{ 2, 3, 4\}$

c- Write a pseudo-code to find the items of the optimal solution.

```
Ks(A[1…n].v.w, Wmax){
    c <- zeros[0…n, 0…Wmax]
    for i<-1…n do
      for j<-1…Wmax do
       c[i,j] = max(c[i-1, w], c[i-1, w - A[i].w] + A[i].v)
k <- j
s <- null
for i <- n…1 do
 if c[i,k] != c[ i-1, k] do
s <- s U {i}
k <- k - A[i].w
return s
}
```

Problem 3

The Longest Decreasing Subsequence problem is defined as follows: Given a sequence of ***n*** real numbers **A[1]... A[n],** determine a subsequence (not necessarily contiguous) of maximum length in which the values in the subsequence form a strictly decreasing sequence.

Example:

The length of the longest decreasing Subsequence in [-1, **2,** -19, **-5**, 4, **-7,** -51, -2, -22, **-13, -15**, 36] is 5.

a- Give the pseudo-code of a Dynamic programming algorithm that solves the Longest Decreasing Subsequence problem.

```
LDS(A[1...n])
  let L[1...n] be an array of length n initialized with 1   → 1
  for i = 1 to n                                            → n
    for j = 0 to i-1                                        → n*i
      if A[j] > A[i] and L[j] + 1 > L[i]                    → n*i*j
        L[i] = L[j] + 1                                     → n*i*j
  return max(L[1...n])                                      → 1
```

b- What is the time complexity of your algorithm? Prove it!

$n(n*i)+2(n*i*j)+2 \leq c*g(n)$

$n^2*n*i+2n*2i*2j+2 \leq n^2 +2n^2+2n^2$

$\leq 5n^2$

$c = 5, n_o = 1$ $\quad \therefore O(n^2)$